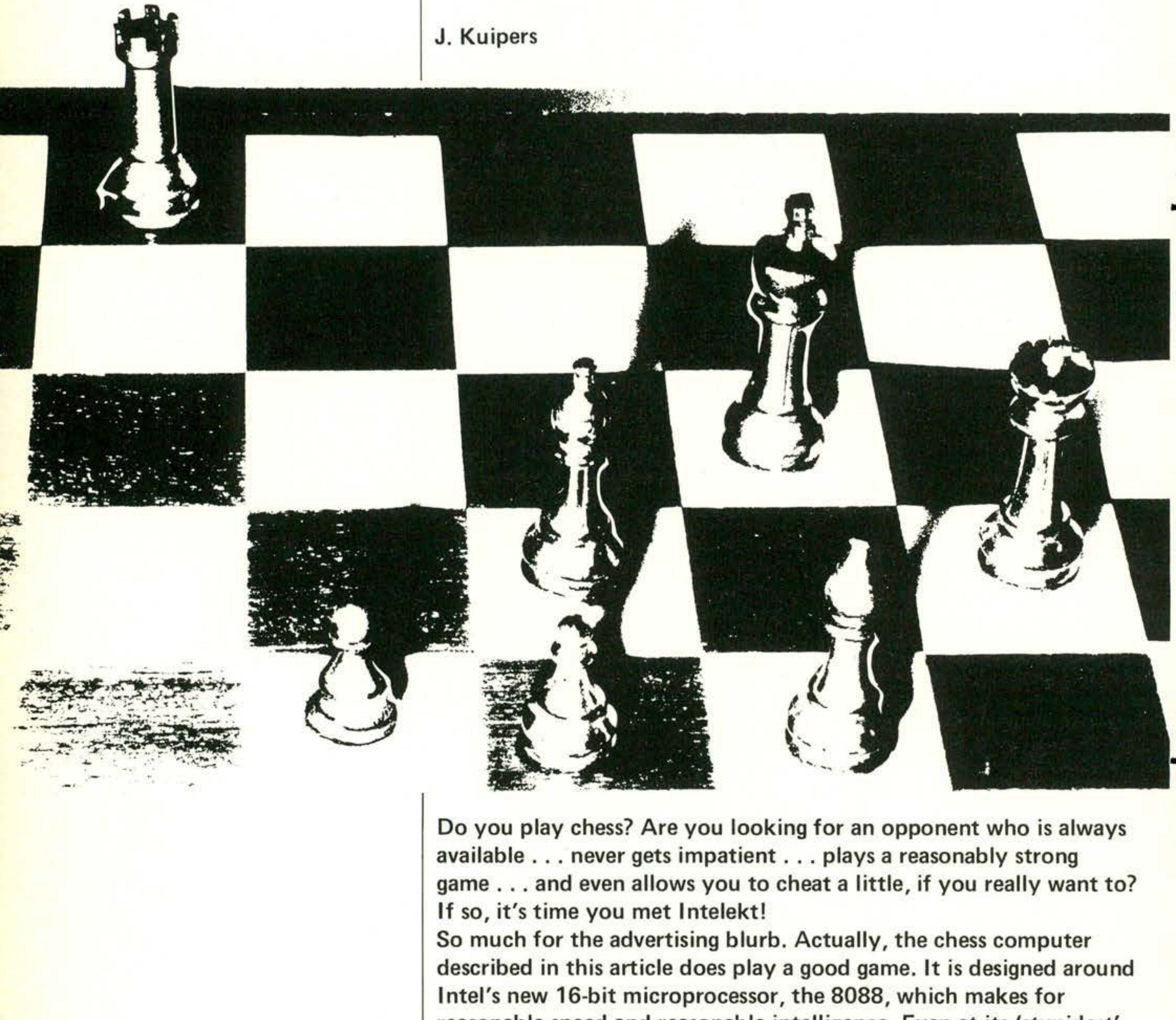
intelekt

a sixteen-bit chess set

J. Kuipers



Do you play chess? Are you looking for an opponent who is always available . . . never gets impatient . . . plays a reasonably strong game . . . and even allows you to cheat a little, if you really want to? If so, it's time you met Intelekt!
So much for the advertising blurb. Actually, the chess computer described in this article does play a good game. It is designed around Intel's new 16-bit microprocessor, the 8088, which makes for reasonable speed and reasonable intelligence. Even at its 'stupidest' level of play (25 seconds per move) it will make a worthy opponent for many chess enthusiasts. At level three (out of eight), it thinks for five minutes or so per move — and provides what we considered a challenging game. Obviously, this evaluation is based to a large extent on our own chess skills. You can judge for yourself: some examples of actual games are included, with comments. If you feel that we played a stupid game, there are still five more intelligence levels to go; on the other hand, if the games look complicated, Intelekt would love to challenge you!

Chess computers are no longer a novelty. This is surprising, when you think of it: a few years ago, it seemed unlikely that even big commercial computers could be taught to play a reasonable game! By now, however, you can buy domestic versions for anywhere between £ 20 and £ 500. By and large, the 'good' machines cost anything from £ 200 up; unfortunately, however, 'intelligence' is not always proportionate to price.

To really determine the best value for money, you would have to play several games against all the available chess computers. We have yet to find somebody who has done this! As 'second-best evaluation', you can either play the machines against each other (with the risk that both play stupid moves without realising it) or else try them out on chess problems. The latter course, in particular, seems very popular for 'comparative reviews' in magazines. In our opinion, this is a very second-rate approach: the fun and challenge in chess is not in solving 'mate in three' (when 'mate in four' is easy); the idea is to manoeuvre your opponent into a position where you can 'mate' him! In other words, the fun is in playing the game — not in ending it.

What is all this leading up to? Quite simple: if you want to know how 'good' Intelekt is in comparison with other chess computers . . . we don't know! (Hurriedly:) But we get the impression that it's pretty good. We tried it out on chess problems that have been used in reviews. Where there was one obvious 'correct' move, Intelekt found it — often even at level 1. Where there was an obvious move that led to mate in four or five and an unexpected one that gave mate in three, it invariably selected the 'obvious' move. However, we played games against a few commercial machines that scored highly in reviews, and found them rather unexciting; we played against Intelekt and it was good fun.

To sum up its strong points in a few nutshells:

- it is easy to set up any position (even halfway through a game);
- illegal moves are not accepted;
- it knows all the rules of the game; castling, for instance, is obviously taken into account as a 'possible move';
- it can play either black or white (or even both sides!);
- it knows the value of sacrificing a piece to gain positional advantage: not only will it ignore this kind of 'sacrifice', it will even propose them where this seems worth trying;
- it plays a good game. This, in our opinion, is what counts.

Who or what is Intelekt? He (or it) is an electronic circuit containing a microprocessor and a chess program (in ROM), with an input/output that must be connected to a 'terminal' — the 'Elekterminal' (Elektor, November/December 1978), for instance. You make your moves by entering them on the keyboard of the terminal; Intelekt answers by displaying the board, his moves and comments (!) on a TV screen, via the same terminal. In other words, Intelekt is a brain; to speak to him and receive his replies you also need a 'terminal'.

In this article, we will give a brief description of the 'hardware' that is involved (circuit and printed circuit board) but no indication of the 'software' (the actual program). Instead, we will attempt to give as clear an impression as possible of his chess skills. After all, that is what counts!

The hardware

The complete circuit is shown in figure 1. It is not our intention to discuss it in minute detail, but we will attempt to paint a sufficiently clear overall picture.

To start with the '16-bit brain' (the 8088): this microprocessor can run in either 'minimum' or 'maximum' mode, depending on the logic level at pin 33. As the words indicate, maximum mode is intended for large systems and minimum mode for little ones. Intelekt belongs in the latter category. As explained in the supplement on 16-bit microprocessors, the 8088 produces the bus control signals itself when it is set to minimum mode; in maximum mode, a further IC would be needed to control the (more extensive) bus.

Inside the CPU itself, data is handled as 16-bit 'words'. However, the data bus that connects it to the outside world is only 8 bits wide. This means that each 16-bit word must be cut into two 8-bit bytes before it can be put on the data bus. For obvious reasons, these two chunks of data are transmitted one after the other — not simultaneously . . . In other words, they are 'time multiplexed'.

In actual fact, things are even more complicated. When Intel introduced the 8085 (a 'normal' 8-bit microprocessor), they used a single set of pins for a multiplexed address/data bus. Now, in the 8088, they've used the same system: the lowest eight address bits also appear on what we have so far called the data bus. This saves pins, making for a smaller and cheaper IC package, and the information is still available at the exact moment that it is required. First the address bits, obviously (the ALE pin indicates that a valid address is being output); then the data, in two 8-bit chunks.

1

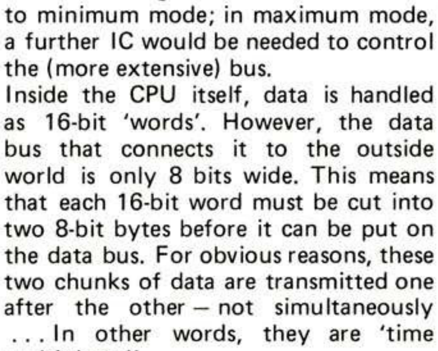


Photo 1. Some of the main control signals, as they appear on the screen of a normal oscilloscope.

Having saved seven pins (the eight multiplexed pins are saved, but ALE must be added), any normal designer immediately starts wondering what he can do with them. Apparently, Intel designers are no different. On the 8085, they used the pins for interrupt signalling; now, in the 8088 we find the address range has been extended to 1 Mbyte (one *million* bytes of memory!).

If the special Intel memory ICs are used, seven tracks can also be saved on the printed circuit board. However, we decided against this; instead, the data and address buses are separated by means of an octal latch (IC3), so that the address information is always available and normal memory ICs can be used.

If all this seems complicated, take a look at photo 1. This shows a group of signals, as they would appear on a 'normal' oscilloscope (not 'cleaned up' by a logic analyser). The upper line is the clock, ticking over at 5 MHz (!); all further timing is derived from this. At point ①, the processor has transmitted the new address information. One of the address/data outputs (ADO) is shown as the second trace. This is followed immediately by the ALE pin (third trace) going high, indicating that a valid address is now present at the output of the CPU. The corresponding output from the address latch (IC3) is shown as the fourth line from the top; as can be seen, each time ALE goes high this output assumes the same level as that on the ADO line, and holds it until the next ALE pulse appears.

If the processor now intends to 'read' data, it sets pin 32 ($\overline{RD}$) to a low logic level as can be seen in the fifth trace on the photo. When the data is to be read from EPROM, the correct chip has already been selected by the preceding address cycle. The $\overline{RD}$ line is connected to the $\overline{OE}$ pin (output enable) of both EPROMs, so the selected memory chip will now put the desired data on the bus (at ③ on the second trace). The processor 'reads' this data and immediately returns the $\overline{RD}$ pin to a high logic level. It can now put the next address on the bus, after which the whole cycle is repeated.

When reading from RAM, the basic principle is the same. However, this type of memory does not include an 'output enable' pin, so the read (or write) signal is included in the 'chip select' logic (CS).

Writing into RAM is similar to reading. As before, the first step is to select the address. Then, immediately after the negative-going edge of the ALE

1

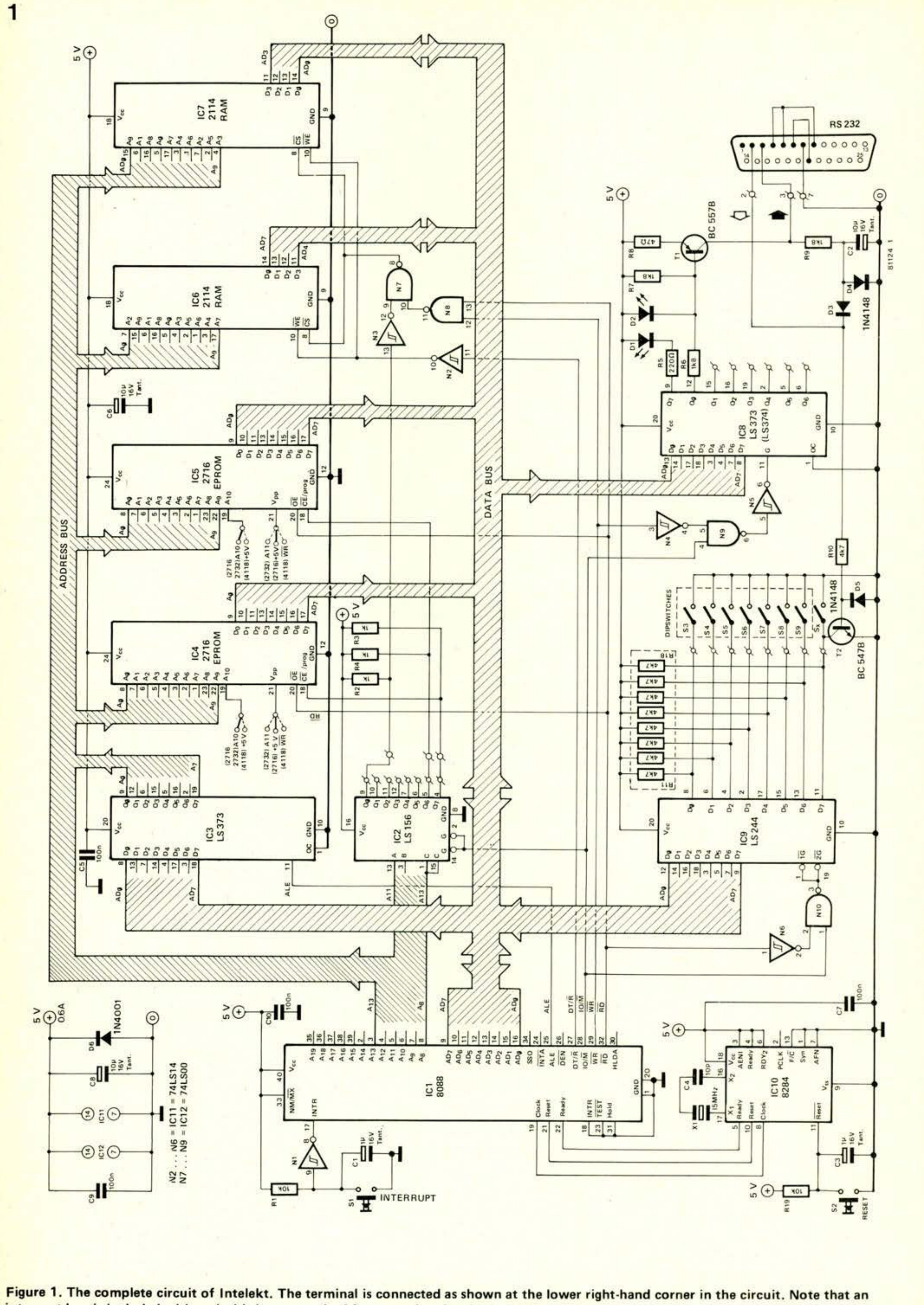


Figure 1. The complete circuit of Intelekt. The terminal is connected as shown at the lower right-hand corner in the circuit. Note that an interrupt key is included, although this is not required for operating the chess computer.

pulse, the processor puts the data onto the AD lines (③). It then sets $\overline{WR}$ at logic 0 (the sixth trace on the photo) to indicate that the data is valid. This 'write' signal is combined with the address information; the correct address is selected and the data is stored in RAM. The data on the bus remains valid for the complete duration of the write pulse.

So far, so good — but how is the RAM to know whether it is to transmit or receive data? This is where the DT/R signal comes in ('data transmit/receive'; the lower trace in the photo). As the address information goes out, this pin is set to logic 1 for a write cycle, or to logic 0 for read. It is passed through an inverter to drive the $\overline{WE}$ (write enable) inputs to the RAMs.

One address in a million

Although the 8088 can handle over one million addresses, Intelekt only needs a good 16,000. Obviously, things would tend to get confusing if several 'chips' started to 'talk' at once. At any given moment, the CPU should only be in contact with one memory IC, and this is where the address decoder (IC2) comes in. This IC monitors three of the address lines (A11 . . . A13) and converts them into eight chip-select signals. Depending on the address range indicated, one of these chip-select signals goes to logic 0 and the corresponding memory IC can communicate with the CPU via the data bus. If the processor wants to talk to an input/output IC, it sets the $IO/\overline{M}$ line to logic 1, deactivating the address decoder.

The address decoder has open-collector outputs. This simplifies matters if several outputs are to be combined — for instance when using larger EPROMs in some future updated version. Each output defines a 2 k address block, so two of these blocks would have to be combined (by connecting the two corresponding pins of the address decoder together) if a 4 k EPROM, type 2732, is to be used.

Although the RAM chip used (the 2114) is only a 1 K type, there is no reason why it should not be allocated its own 2 K block of addresses. (Note that the two 2114s each take care of four data bits; together, they form the 1 K x 8 memory). As shown in figure 2, the RAM is located at the lowest memory addresses — from 00000 to 003FF. Since it is enabled during the complete 2 K block, a duplicate RAM area appears from 00400 to 007FF. In other words, two different addresses define each RAM memory cell.

The input/output (I/O) chips IC8 and IC9 don't need an address decoder. All I/O write instructions enable IC8, via the combined $IO/\overline{M}$ and $\overline{WR}$ signals; similarly, all I/O read operations refer to IC9.

Allocating the addresses

One might assume that EPROM and RAM could theoretically be located anywhere in memory. In practice, this is not quite true. When the processor is reset, it starts to run a program from address FFFF0 on. To make sure that there is a program there, it is advisable to locate an EPROM in this final address block.

Furthermore, after an interrupt, the processor goes looking for an 'interrupt vector' (this is the address where the corresponding interrupt routine is located) at one of the lower memory addresses. Since it is useful to be able to change these addresses, RAM must be located in the lowest address block. Even though Intelekt doesn't actually make use of the interrupt facility, it was decided to locate the memory at the 'normal' addresses. This leads to the situation shown in figure 2: RAM (with its duplicate) in low memory, and two blocks of EPROM (4 k in all) at the top.

All this may seem quite reasonable, until you start thinking it over. EPROM is at the top and RAM at the bottom of a one *mega*-byte address range — but the address decoder is only defining eight 2 K blocks of memory! How can 16 K be equal to 1 M?

Since we are now tossing out K's and M's at the rate of one or two in each sentence, it is perhaps a good idea to digress briefly and explain what they signify. Using a single address line, you could distinguish between two addresses. With two lines, you get an 'address range' of four addresses; three lines define eight addresses, and so on. By the time you get up to ten lines, you find that you can distinguish between 1024 adresses. This is referred to as a '1 K block'. Since it is slightly more than one thousand, we use a capital K. It's rather like the difference between Imperial and US gallons: they're both gallons, but one is slightly more than the other. Similarly, the 1 Mbyte address range of the 8088 is slightly more than one million addresses: 20 address lines define 1,048,576 addresses.

Back to our 'problem': how can 16 K be equal to 1 M? Fourteen address lines define a 16 K block; of these lines, the highest three (A11 . . . A13) go to the address decoder. All higher address lines are simply ignored! This means that the address decoder can't see any difference between addresses 0000, 04000, 08000 and so on. This can be seen in table 1, where the actual bits on the various address lines are shown for these addresses. Reading from left to right, these bits are used as follows:

- A19 . . . A14 are ignored. They can have any value, without making any difference to the actual memory location that is selected.
- A13 . . . A11 go to the address decoder. They define eight 2 K blocks; the highest two enable the EPROMs, and the lowest 2 K block is for the RAM.
- A10 . . . A0 define the 2048 addresses in each 2 K block. In the lowest (RAM) block, A10 is also ignored; this means that the same RAM is addressed in both the first and second 1 K block (these are referred to as 'RAM' and 'RAM duplicate', respectively).

The answer to the 'problem' should now be clear: the basic 16 K address range is simply duplicated 64 times in the total 1 Mbyte range, as shown in figure 2. After reset, the processor looks at address FFFF0. The address decoder looks at lines A11 to A13, finds them all at logic 1, and enables the first EPROM. Exactly the same result would be obtained if the processor tried to address 'location 03FF0'.

Interface

Communication with the outside world runs over a simple RS-232 interface (T1 and T2). The 'receiver' is a single transistor that converts the input signal levels to TTL logic levels:

$-12 \ldots -5$ V = logic 1 $\rightarrow +5$ V;
$+5 \ldots +12$ V = logic 0 $\rightarrow 0$ V.

Diode D5 protects the transistor when the input signal swings negative.

The 'transmitter' end is also a single transistor. This one operates as a voltage-to-current converter, which

2

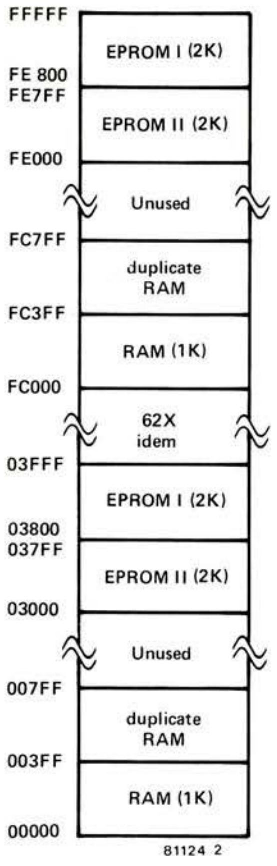


Figure 2. The 'memory map'. Although the theoretical address range is 1 Mbyte, it actually consists of 64 identical 16 Kbyte sections.

3

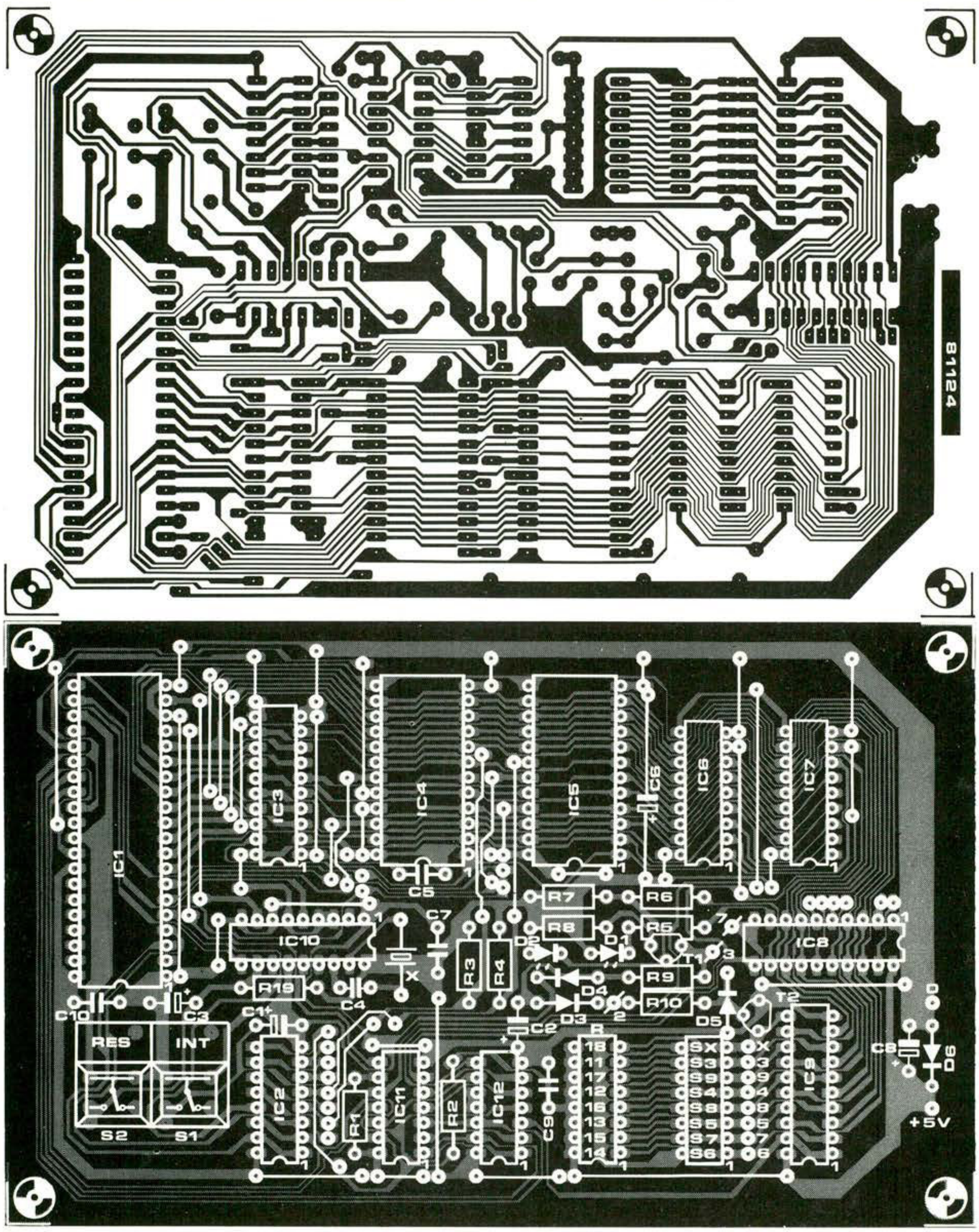


Figure 3. The printed circuit board. Particular care should be taken with the wire links: 43 are required, at least. If the interrupt key is omitted, a further wire link is required to connect what would otherwise have been the centre contact pins, as shown. The same applies for the reset key, if this is mounted off the board or replaced by a type that does not contain this internal connection. Where several wire link options are possible, for other memory ICs, only the correct link is shown.

Parts list

Resistors:
R1,R19 = 10 k
R2,R3,R4 = 1 k
R5 = 220 Ω
R6,R7,R9 = 1k8
R8 = 47 Ω
R10 = 4k7
R11 . . . R18 = 8 x 4k7 (or 16-pin DIP resistor network)

Capacitors:
C1,C3 = 1 μ/16 V Tantalum
C2,C6,C8 = 10 μ/16 V Tantalum
C4 = 10 p
C5,C7,C9,C10 = 100 n

Semiconductors:
D1 = LED
D2 = LED (red)
D3 . . . D5 = 1N4148
D6 = 1N4001 or surge diode (TVS 505, for instance)
T1 = BC 557B
T2 = BC 547B
IC1 = 8088
IC2 = 74LS156
IC3,IC8 = 74LS373
IC4,IC5 = 2716 EPROM 450 ns
IC6,IC7 = 2114 RAM 450 ns
IC9 = 74LS244
IC10 = 8284
IC11 = 74LS14
IC12 = 74LS00

Miscellaneous:
S1 = digitast switch or wire link (see text)
S2 = digitast switch
S3 . . . S9, Sx = 14- or 16-pin DIP switch (or wire links, see text)
X1 = 15 MHz crystal small size HC-18/U

automatically makes it short-circuit proof. LED D2 is used to set the base voltage — it will barely light, since the current through it is only 2 mA. To meet the RS-232 standard, a negative output voltage is also required. Since Intelekt only uses a positive supply, a little trick is used. The *input* signal, coming from the terminal, printer or whatever, swings between positive and negative levels. This signal is rectified (by D3, D4 and C2) to provide the negative 'supply' for the output.

A second output from IC8 drives LED D1. This LED flashes on and off when the chess program is running. Regular and fairly rapid flashes indicate that he is waiting for you to enter data; slower flashes (corresponding to the depth of the 'search') will appear when he is thinking.

Of the eight inputs to IC9, one is used for the RS-232 input. The others can be connected to a DIP switch; this is a unit that contains seven or eight miniature switches, and can be plugged into a normal IC socket. Note that, if an eight-switch version is used, the lower switch should not be closed — otherwise it would short the RS-232 input to ground!. Three of the switches set the baud rate, as listed in table 2. Obviously, for a fixed baud rate (=transmission speed to and from the terminal) wire links can be used instead of the switches.

Construction

The printed circuit board is shown in figure 3. To keep the cost down to a reasonable level, it was decided to use a single-sided board. This does lead to a larger number of wire links. There are 43 in all, and it's worth counting them before switching on for the first time!

The possibility of future extensions was also considered, and some points were brought out even though Intelekt doesn't use them. However, this does not mean that the board can be used as the basis for an extensive system: the bus is not buffered, and the addresses are not fully decoded. The only possible extensions we have in mind are the use of other EPROMs (or ROMs) with a 4 K range, and extension of the RAM area by substituting a 4118, say, for one of the EPROMs. In general, the flexibility that the board offers is only intended to facilitate its use in other small-system applications.

The main wire links to watch in this connection are:

- those at each EPROM socket: they determine whether a 2716, 2732 or 4118 can be used — for Intelekt, the '2716' link is used.
- the chip enable (CE) inputs to the EPROMs and RAM are connected to the address decoder as required; for Intelekt, EPROM 1 is driven from output 7, EPROM 2 from output 6 and RAM from output 0.
- the DIP switch (or wire links) set the baud rate; it is set according to table 2.

Table 1.

Address (hexadecimal)	Address (binary)																			
	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0 0 0 0 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 3 F F F	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0 4 0 0 0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 8 0 0 0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F C 0 0 0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F F F F 0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0

Table 1. Of the 20 address lines, only the lower 14 are actually used. A11 . . . A13 are passed to the address decoder, to determine which memory chip must be enabled. The logic levels of the higher address bits (A14 . . . A19) are irrelevant.

Table 2.

Baud rate	S_6	S_5	S_4
9600	0	0	0
4800	0	0	1
2400	0	1	0
1200	0	1	1
600	1	0	0
300	1	0	1
150	1	1	0
110	1	1	1

Table 2. The baud rate (transmission speed in bits per second) is determined by the setting of three of the switches in the DIP switch block. If a fixed rate is sufficient, wire links can be used instead.

4

```
8  RR   NN   BB   QQ   KK   BB   NN   RR

7  PP   PP   PP   PP   PP   PP   PP   PP

6  :::       :::       :::       :::

5       :::       :::       :::       :::

4  :::       :::       :::       :::

3       :::       :::       :::       :::

2  P    P    P    P    P    P    P    P

1  R    N    B    Q    K    B    N    R

   A    B    C    D    E    F    G    H
                                    81124 4
```

Figure 4. The initial board position, as it will appear on the screen. The single letters (R, N, B, . . .) are white pieces; the letter pairs are black; the dots (:::) indicate a vacant white square.

Two digitast switches are also mounted on the board. Note that, on this type of switch, the centre contact is brought out onto two pins. *This fact is used on the board to connect 'supply common' to a whole section of board. If other types of switches are used, or if the switches are mounted off-board, two further wire links will be required at this point!*

Communicating with Intelekt

As explained earlier, Intelekt is controlled via the keyboard of a terminal and he 'talks back' by means of the associated display (TV screen or printer). To get some idea of how this works in practice, assume that Intelekt is connected to the Elekterminal.

After switching on, first operate the Reset key on the chess computer itself — note that this is the only command that is not entered via the Elekterminal keyboard. Intelekt will respond by displaying the following message:

TINY CHESS VI.O
LEVEL IS 1 CHANGE TO _

You can now enter a '1', followed by Carriage Return — the 'why' of this will be explained later on. The chess board will now appear on the screen, in the initial position as shown in figure 4. The single letters (R, N, B, . . .) stand for the white pieces, the letter pairs (RR, NN, etc) are black pieces and the dots (:::) are empty white squares on the board. Intelekt then asks for your first move:

01W:

To enter your move, key in:
- the square containing the piece that is to be moved;
- a space;
- the square to which the piece is to be moved;
- Carriage Return.

Intelekt will now check whether or not you have entered a legal move (if not, he will request a new entry) and then proceed to calculate his response. Initially he will invariably find a response in his 'book of standard openings' and reply immediately. Later on, when he has to start thinking out his moves, the response time can vary from 25 seconds (lowest level of skill) up to several hours (highest level). Having worked out his move, Intelekt will print it on the screen, and immediately display the new position on the board. The result so far could look like this (bold type: your moves):

TINY CHESS VI.O
LEVEL IS 1 CHANGE TO **1**
(initial board situation)

01W: **e2 e5 (CR)** – an illegal move, so:
01W: **e2 e4 (CR)**
01B : c7 c5
(new board situation)
02W: - waiting for your next move.
If you notice a typing error before entering Carriage Return, it is possible to correct this by operating 'Backspace'. However, if it was a legal move and you have already typed Carriage Return, there is no easy way to correct it. If you really want to cheat, it is possible to enter an illegal move, provided you terminate it with 'Line Feed' instead of 'Carriage Return'.

When it is your move, you can also enter one of the following instructions:
Control X: change players. You now play black; he responds by printing 'my move'. After he has made his move, you can change back to playing white by again entering Control X.
Control A: autoplay. He plays both sides!
Control N: to set the 'number' of the level that you want to play. He responds with 'Level is 1 change to _' (assuming that you were at level 1); you can then enter any number between 1 and 8, followed by Carriage Return. Level 1 is the easiest and level 8 the most difficult. We found level 3 to be a good compromise between response time (about 5 minutes, on average) and skill.
Control C: change board mode. Intelekt responds with a dash prompt: '–'; you can now enter one of several commands:

- erase the board (remove all pieces) by entering Control E,
- change any square, as follows:
  - enter the number of the square (b5, say); Intelekt responds by printing what is on that square;
  - if desired, update the square by entering either a colon (:) to empty the square, or a single letter (K, Q, R, B, N or P) for the corresponding White piece, or two letters (KK, QQ etc.) for a Black piece;
  - enter a Space to step to the next square (whether or not you have updated the preceding one);
  - enter Carriage Return when a sequence of squares has been updated. Intelekt again responds with a dash prompt, waiting for you to enter a new square.
- after editing the board, return to normal mode by entering Carriage Return. It may be worth noting that Intelekt will refuse to play unless there are a Black and White king on the board . . .

All the commands listed above can only be entered when it is your turn. So what do you do when Intelekt is thinking? You can 'interrupt' him by entering *Break* or *several spaces*. This has the same effect as the Control N instruction described above: you can change the level of play. By entering a lower level (level 1, say) you can ensure that it will be your turn within half a minute; at that point you can of course enter any command.

Reset: This resets the board and program for a new game.

Special moves

Entering 'normal' moves was explained above. For those who are not so familiar with the numbering of the squares (A to H left to right, and 1 to 8 from bottom to top), each board print-out includes these letters and numbers. There are also a few special moves: castling, en passant taking of a pawn, check and pawn promotion. All of these possibilies are known to Intelekt. They are dealt with as follows:
Castling: only enter the move for the king. Intelekt will interpret this correctly, check whether or not it is permissable and then move both king and rook accordingly.
En-passant: this move is not as well known as it ought to be. To put it in a nutshell: when a pawn is initially moved up two squares, passing a square that is attacked by a pawn, it can be taken at the next move by that pawn. As an example, assume that Black has a pawn on b4. If white moves a pawn a2-a4, Black can take it immediately by moving b4-a3. To execute this move, you would simply enter 'b4-a3'; Intelekt will know what is meant.
Check: Intelekt will print a warning when it places you in Check (or Checkmate); it then rejects any move that doesn't remove your king from check. Stalemate is also recognised.
Pawn promotion: It is presumed that when you promote a pawn, you want a queen; and Intelekt calculates its moves on this basis (a minor 'blind spot'). If you want anything else, this can be obtained via the 'change board' mode.

A few games

Three complete games are given in tables 3 . . . 5. In the first, fairly straightforward game, Intelekt played black; in the second, he played white. In the third game, Intelekt again played black; furthermore, in this game white made a deliberate effort to 'draw out' the machine as far as possible before striking back – too late, as things turned out . . .
Obviously, it would take up too much space to examine each game in great detail. However, if you are interested in playing out each game according to the moves listed in the corresponding table, we will attempt to pick out the interesting highlights.

Game 1

The first two moves were according to his opening 'book': Black's response was immediate. White's third move *(g2-g3)* put a stop to this; from now on, Intelekt must start thinking for himself . . .
After some manoeuvering and minor skirmishes, White's move *13. f2-f4* was a deliberate attempt to make things complicated. If, on the next move, White takes one of Black's pawns (*f4xe5* or *f4xg5*) the rook on f1 would attack Black's queen and things would start to happen . . . Black can't take the pawn by playing *e5xf4*, and *g5xf4* opens a lot of interesting possibilities.
In fact, things developed nicely. Then, at the 17th move, White was faced with the choice: *Nd5-f4* or attempt to break up Black's central pawn formation? He chose the latter option, but it didn't quite work out as planned . . . Not yet, anyway.
Moves 20 and following may seem rather strange at first sight. *20 Ra1-d1* is safe enough: Black can't play *Bh5xd1*, since this would be followed by *Qf2xf7 mate*! To remove this threat, Black tried *f7-f5*. This led to the loss of a pawn, and White even got the opportunity to continue the

Table 3.

1.	e2-e4	c7-c5
2.	c2-c4	e7-e5
3.	g2-g3	d7-d6
4.	Bf1-g2	a7-a6
5.	Nb1-c3	Nb8-c6
6.	d2-d3	Nc6-d4
7.	Ng1-e2	Bc8-d7
8.	O - O	Bd7-g4
9.	b2-b3	Nd4xe2†
10.	Nc3xe2	Qd8-f6
11.	Bc1-b2	g7-g5
12.	Qd1-d2	Bg4-e6
13.	f2-f4	g5xf4
14.	Ne2xf4	Be6-g4
15.	Nf4-d5	Qf6-d8
16.	Qd2-f2	Bg4-h5
17.	b3-b4	c5xb4
18.	d3-d4	b4-b3?
19.	a2xb3	Bf8-g7
20.	Ra1-d1	f7-f5
21.	Qf2xf5	Bh5-g6
22.	Qf5-f2	Bg6-h5
23.	g3-g4	Bh5-g6
24.	h2-h4	Ng8-h6
25.	g4-g5	Nh6-g8?
26.	d4xe5	Bg6-h5
27.	e5xd6	Bg7xb2
28.	Qf2xb2	Bh5xd1
29.	Qb2xh8	Bd1xb3
30.	Qh8xg8†?	Ke8-d7
31.	Qg8xh7†	Kd7xd6
32.	Nd5-b6?	Qd8xb6†
33.	Kg1-h1	Bd3xc4?
34.	Rf1-f6†	Bc4-e6
35.	Rf6xe6†!	Kd6xe6
36.	Qh7-h6†	Ke6-f7
37.	Qh6xb6	. . . . .
	"I knew that"	
	. . . . .	Ra8-h8
38.	Qb6-f6†	Kf7-g8
39.	g5-g6	Rh8xh4†
40.	Qf6xh4	Kg8-g7
41.	Qh4-h7†	. . . . .
	"GRRR"	
	. . . . .	Kg7-f6
42.	g6-g7	b7-b5
43.	g7-g8 (Q)	. . : . .
	"I knew that"	
	. . . . .	"I give up"

Table 3. The first game, with Intelekt playing black.

'mopping up' action in the centre (moves 26 and following). During a momentary lull in the battle, Black decided it would like to take a pawn (*29 . . . Bd1xb3*), leaving White with so many options that he didn't know which to choose! *Nd5-c7*†, followed by *Qh8xg8*†, might well win a rook. On the other hand, *Qh8xg8*† seems quite promising already. Or *Qh8xh7*? Or *Rf1-b1*? Or *e4-e5*? The game had already lasted three hours, so White decided to pick one alternative at random . . .

32 Nd5-b6 was a mistake, pure and simple. The idea was to pin things in that corner *(Qd8xb6* was to be followed by *Rf1-f6*, and if Black tried to save this rook, White could follow up with *Rf1-d1*), but White forgot that Qb6 gives check! Amazingly, Intelekt offered this option one move later — apparently assuming that *Bc4-e6* would be a sufficient answer. It wasn't, as moves 35 . . . 37 show.

What followed was just a fairly brutal end-game, punctuated by various comments from Intelekt.

Game 2

By coincidence, this game (with Intelekt playing White) developed along the same initial lines as the previous one. As before, Black's third move (*g7-g6*) put an end to Intelekt's use of his 'opening book'. From here on, the game progressed in a fairly conventional manner, until things started to happen around the tenth move. After the dust had cleared (at the fourteenth move), Black was two pawns up and had a considerable positional advantage.

As things seemed to be developing in Black's favour, White obviously decided that it would be wise to exchange queens (moves 18 and 19). There didn't seem to be much point in avoiding this exchange . . . From here on, things again developed slowly but surely. For the fun of it, Black tried a little trap at move 31 (*e5-e4*†): *32 Kd3xe4* would have been followed by *Nc4-d2*†, winning a rook. As expected, White saw this trap and avoided it.

Some further manoeuvring (up to and including move 39) led to a position where most of the remaining pieces were tied up in one corner, leaving the game essentially as a standard two-pawn-against-one-pawn end-game. Surprisingly, White gave the impression that it was going to chase away Black's rook with its king, so Black decided to simply promote the pawn without taking further precautions (move 43). However, the White king came back — prolonging the agony. By move 49, the end result was clear (even to Intelekt); in sheer desperation he tried sacrificing his bishop. To no avail.

If you think it looks easy to 'beat the monster', the following may prove interesting. At move 19 in this game, the situation was as shown in figure 5. At this point, Black seriously considered playing *19 . . . Qa5-c3*. For various reasons (mainly that it didn't 'feel' right), he chose the alternative given earlier (*e4xd3*). However, since it is very easy to set up any position when playing against Intelekt, the second-best choice was also tried later on. The results are shown in the continuation of table 4 . . .

The key move was *23 Qb5-e8*†, leading to mate in four moves — or so it seems. However, when it came to move 25 he played *Qe7xh7*† instead of *Rh1xh7* followed by mate (. . . *Qc1xc4*†*, 26 Nd2xc4, Ng4-f6, Qe7-g7 mate*). Prolonging the agony? Or was this 'beyond his horizon'? Or did he see greater danger in . . . *d4-d3*†, which opens new possibilities for harassment by Black? However this may be, Black now has a small reprieve; he even gets a chance to set a small trap: *29 . . . Qc1-a3*. It seemed conceivable that White might play *Qd4-c3*, which could be followed by *Ng4xf2* — winning a rook). However, no such luck. Even though it took some time, Intelekt succeeded in ending the game. To be quite honest: his end-game could do with some improvement! He gets there in the long run, but it could often be a lot shorter . . .

Table 4.

1.	e2-e4	e7-e5
2.	c2-c4	c7-c5
3.	d2-d3	g7-g6
4.	h2-h3	Bf8-g7
5.	h3-h4	Ng8-f6
6.	g2-g3	d7-d6
7.	Qd1-a4†	Nb8-c6
8.	Bc1-g5	O - O
9.	h4-h5	Qd8-b6
10.	h5-h6	Bg7xh6!
11.	Bg5xh6	Qb6xb2
12.	Bh6xf8	Qb2xa1
13.	Bf8xd6?	Qa1xb1†
14.	Qa4-d1	Qb1xa2
15.	Ng1-f3	Nf6-g4
16.	Nf3-d2	f7-f5
17.	Bd6xc5	f5xe4
18.	Qd1-b1	Qa2-a5
19.	Qb1-b5	e4xd3
20.	Qb5xa5	Nc6xa5
	"I knew that"	
21.	Bf1xd3	Bc8-e6
22.	Bc5-b4	Na5-c6
23.	Bb4-c5	Ra8-c8
24.	Nd2-f3	Nc6-a5
25.	Bc5xa7	Na5xc4
26.	Nf3-g5	Nc4-b2
27.	Ke1-d2	Be6-c4
28.	Bd3xc4†	Nb2xc4†
29.	Kd2-d3	h7-h5
30.	Ng5-e6	b7-b6
31.	Rh1-f1	e5-e4†
32.	Kd3-d4	e4-e3
33.	f2xe3	Ng4xe3
34.	Rf1-f3	Rc8-e8
35.	Ne6-c7	Re8-e7
36.	Nc7-b5	Re7-d7†
37.	Kd4-e4	Nc4-d2†!
38.	Ke4xe3	Nd2xf3
39.	Ke3xf3	Rd7-b7
40.	g3-g4	Kg8-h7
41.	g4xh5	g6xh5
42.	Kf3-f4	Kh7-g6
43.	Kf4-e5?	h5-h4
44.	Ke5-f4!	Kg6-h5
45.	Kf4-f3	Kh5-g5
46.	Kf3-g2	Kg5-g4
47.	Kg2-h2	h4-h3
48.	Kh2-h1	Kg4-g3
49.	Ba7xb6!	Rb7xb6
50.	Nb5-c3	Rb6-e6
	"I give up"	

Mistakes are not permitted. This is illustrated by the following alternative play from move 19:

19.	Qb1-b5	Qa5-c3?
20.	d3xe4	Qc3-c1†
21.	Ke1-e2	Nc6-d4†??
22.	Bc5xd4	e5xd4
23.	Qb5-e8†!!	Kg8-g7
24.	Qe8-e7†	Kg7-g8
25.	Qe7xh7†	Kg8-f8
26.	Qh7-h8†	Kf8-f7
27.	Qh8xd4	Bc8-f5
28.	e4xf5	Ra8-e8†
29.	Ke2-d3	Qc1-a3†
30.	Kd3-c2	Qa3-a4†
31.	Nd2-b3	Re8-c8
32.	Rh1-h7†	Kf7-g8
33.	Qd4-g7†	Mate.

Table 4. In this game, Intelekt played white. From the nineteenth move on, two variations were tried.

5

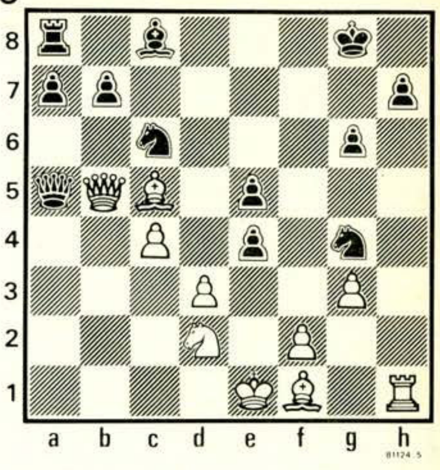


Figure 5. In the second game, this position was reached after White's 19th move. An alternative play was tried from this point.

Game 3

This game proved quite interesting. Intelekt played Black, and White deliberately attempted to 'draw him out'. As it proved, giving Intelekt an advantage is fatal — even at level 3!

The surprises started at move 7: *Nd4-f3*†! Suicide? Not at all, as the continuation proves. To compound the misery, White's thirteenth move was an out-and-out mistake, and Black's response was immediate. Things start to get hectic at this point, culminating in Black's clincher at the twentieth move: *Bf4-h2*. He didn't take the rook — he was after greater glory. Greeting White's only response (*Rg1xg2*) with the comment 'Dummy!' was adding insult to injury. White now decided to get

vicious, but it didn't help.
What follows is relatively uninspired. White tried a little trap at moves 42 and 43: the idea was to follow up with *44 Bg5-e3*, winning the pawn on h5. It didn't work. By the fiftieth move, White didn't feel like doing any hard thinking. *51 c6-c7* might have been better than *Kd6-e7*; and *52 Ke7-d7* might be better than *Ke7-d8*. However: the result seems a foregone conclusion, no matter what. Moves 57 . . . 60 offer a clear opportunity for a draw, on the basis of repeated moves; however, White would like to see how Intelekt wins this game. It's a disappointment. Even with one rook up, he can't work out a clear strategy. After move 73, White had had enough — and switched to autoplay. After move 79 it seemed a good idea to go to bed, and have a look in the morning to see what had developed. Nothing! He was still playing ring around the roses. So I pulled the plug.
The moral of the story: Intelekt plays a good game, but he doesn't always know how to win in an end-game. This is a fairly common failing with chess computers. Fortunately, it doesn't detract much from their charm: the fun is in fighting the game until there is a clear win for one of the two sides. At that point, it would be normal to give up. Playing on against a human opponent would lead to a fairly quick death, so you don't do it; playing on against Intelekt may lead to an endless game, so you don't do that either.

In conclusion

In earlier articles ('How I beat the monster' and 'Computers and Chess', Elektor January 1979), we examined the operating principles and common failings of chess-playing computers. Basically, Intelekt uses the brute force of a fast 16-bit microprocessor to overcome the shortcomings of a straightforward 'mini-max' procedure. Surprisingly enough, this approach works! For most 'average' human chess players, he presents a good challenge at reasonably short response times. In fact, for most of us poor mortals it is fortunate that his 'creator' didn't add sophisticated short-cuts in the program. It would be just too humiliating to be wiped up by a handful of electronics that only took a few seconds to work out a move!
However, there is room for improvement. We have already discussed the possibilities of improving the end-game and including a survey of pawn promotion to other pieces than a queen. Both would involve additional memory and slower response — as things stand at present. But Intelekt's skills are stored in EPROM — and this can be exchanged, at a later date, for a more sophisticated program! Who knows?
Intelekt is intended as a chess opponent. He doesn't like solving chess problems on his own. So what? Playing chess is fun, but getting someone (or something) else to solve chess problems for you is as pointless as looking for a Scrabble opponent who will solve crossword puzzles. We got to like Intelekt — he has a charm of his own. If you like chess, you should make his aquaintance.

Table 5.

1.	e2-e4	c7-c5
2.	g2-g3	e7-e5
3.	Bf1-g2	d7-d6
4.	b2-b3	Nb8-c6
5.	Bc1-b2	Nc6-d4
6.	Ng1-e2	Bc8-g4
7.	Nb1-c3	Nd4-f3†!
8.	Bg2xf3	Bg4xf3
9.	Rh1-f1	Ng8-f6
10.	d2-d3	h7-h5
11.	h2-h4	Qd8-a5
12.	Qd1-d2	g7-g6
13.	O-O-O	Bf8-h6!
14.	Ne2-f4	e5xf4
15.	Nc3-e2	f4xg3
16.	Ne2-f4	Qa5xd2†
17.	Rd1xd2	. . . . .
	"I knew that"	
	. . . . .	g3-g2
18.	Rf1-g1	Bh6xf4
19.	Bb2xf6	O - O!
20.	Bf6-g5	Bf4-h2!
21.	Rg1xg2	. . . . .
	"Dummy!"	
	. . . . .	Bf3xg2
22.	f2-f4	f7-f6
23.	Bg5-h6	Bh2xf4
24.	Bh6xf4	Bg2-h3
25.	Bf4xd6	Rf8-c8
26.	Rd2-f2	f6-f5
27.	e4xf5	Bh3xf5
28.	Rf2-g2	Rc8-e8?
29.	Bd6xc5	Re8-c8
30.	b3-b4	Kg8-h7
31.	Kc1-d2	Bf5-e6
32.	c2-c4	a7-a5
33.	a2-a3	Be6-h3
34.	Rg2-e2	a5xb4
35.	Re2-e7†	Kh7-h6
36.	Bc5-e3†	g6-g5
37.	Be3xg5†	Kh6-g6
38.	a3xb4	Bh3-g2
39.	Kd2-c3	Rc8-f8
40.	Re7-e6†	Kg6-g7
41.	c4-c5	Rf8-f2
42.	Re6-e5	Bg2-c6
43.	Kc3-d4	Rf2-f1
44.	b4-b5	Bc6xb5
45.	Re5-e7†	Kg7-g6?
46.	Re7xb7	Bb5-c6
47.	Rb7-b6	Ra8-a4†
48.	Kd4-e5	Rf1-e1†
49.	Ke5-d6	Bc6-f3
50.	c5-c6	Re1-d1
51.	Kd6-e7?	Rd1-e1†
52.	Ke7-d8?	Ra4-d4†
53.	Kd8-c8	Re1-e8†
54.	Kc8-b7	Bf3-g2?
55.	Kb7-a7	Rd4xd3
56.	c6-c7†!	Kg6-f5
57.	Rb6-b8!	Rd3-a3†
58.	Ka7-b6	Ra3-b3†
59.	Kb6-a7	Rb3-a3†
60.	Ka7-b6	Ra3-b3†
61.	Kb6-c5	Re8xb8
62.	c7xb8	Rb3xb8
63.	Kc5-d4	Bg2-c6?
64.	Kd4-c5	Bc6-a4
65.	Kc5-d6	Kf5-e4
66.	Kd6-e6	Ba4-c6
67.	Ke6-f6	Ke4-d3
68.	Kf6-g7	Kd3-e2
69.	Kg7-h6	Bc6-f3
70.	Bg5-f4	Rb8-b6†
71.	Kh6-g5	Ke2-d3
72.	Bf4-c7	Rb6-b5†
73.	Kg5-f4	Kd3-e2
Autoplay:		
74.	Bc7-e5	Bf3-d5
75.	Be5-c3	Rb5-b1
76.	Kf4-f5	Rb1-g1
77.	Bc3-d4	Rg1-c1
78.	Kf5-g6	Ke2-d3
79.	Bd4-e5	Rc1-f1
	. . . . .	. . . . .
	. . . .	. . . . .
	. . . . .	. . . . .
148.	Kg6-g5	. . . . .

giving the board situation:

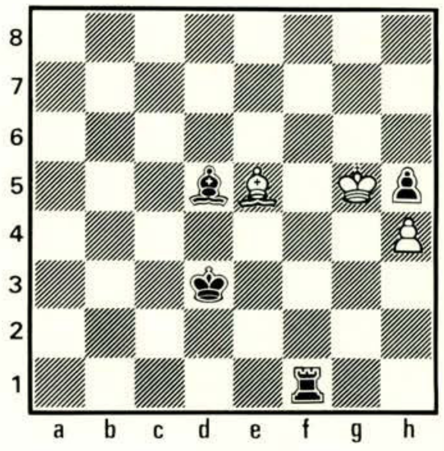


Table 5. The computer plays black, and white tries a deliberately 'passive' game to provoke Intelekt into attacking. This proved fatal . . .

David Levy: 'How I beat the monster'
Elektor, January 1979, p. 140
'Computers and Chess'
Elektor, January 1979, p. 134
'16-bit microprocessors'
Special supplement in this issue.